Code Review Checklist

Project Name: Rocket Launch Simulation
Version: 1.0
Reviewer: Muhammad Awais

Roll No# 233134
Date: 29-Oct-2025

Structure:

Description of Items	Pass(Y/N)	Comment
Does the code completely and correctly implement the design?	N	Empty or incomplete branches in SpaceDemo.java (D-018).
Does the code conform to any pertinent coding standards?	N	Naming conventions and indentation not consistent.
Is the code well-structured, consistent in style, and consistently formatted?	N	Inconsistent spacing, capitalization, and method organization.
Are there any uncalled or unneeded procedures or unreachable code?	Y	Unused methods and dummy object “leftover” created.
Are there any leftover stubs or test routines in the code?	Y	Empty finalize () and placeholder methods.
Can any code be replaced by reusable library functions?	N	Custom loops and retry logic could use standard Java utilities.
Are there repeated code blocks that could be condensed?	N	Duplicate loop and exception handling patterns.
Is storage use efficient?	N	Unnecessary object creation in SpaceDemo.java.
Are symbolics used rather than magic numbers?	N	Hardcoded constants: 3, 42, 1000.
Are modules excessively complex and should be split?	N	Rocket.java mixes multiple responsibilities.

Documentation:

Description of Items	Pass(Y/N)	Comment
Is the code clearly and adequately documented with maintainable commenting style?	N	Many empty comments (`/** */`) and missing headers.
Are all comments consistent with the code?	N	Comments do not reflect actual code logic.

Variables:

Description of Items	Pass(Y/N)	Comment
Are variables properly defined with meaningful, consistent names?	N	Violations in naming (D-019, D-020).
Do assigned variables have proper type consistency or casting?	N	Type mismatches found (D-002, D-004).
Are there redundant or unused variables?	Y	“leftover” object unused.

Style:

Description of Items	Pass(Y/N)	Comment
Does the code follow the style guide for this project?	N	Mixed naming and inconsistent indentation.
Is the header information for each file descriptive?	N	Missing or incomplete package and file-level comments.

Is there an appropriate amount of comments?	N	Minimal comments; unclear code flow.
Is the code well-structured typographically and functionally?	N	Logic spread unevenly; poor readability.
Are variable and function names descriptive and consistent?	N	Names violate conventions.
Are magic numbers avoided?	N	Several found (42, 3, 1000).
Is there dead/unreachable code?	Y	Unused methods and imports.
Is any assembly or low-level code removable?	N/A	None present.
Is the code too tricky or hard to follow?	Y	Logic convoluted in Rocket.java.
Is the code self-explanatory?	N	Requires author clarification.

Architecture:

Description of Item	Pass(Y/N	Comments
Is any function too long?	N	Methods short but unclear.
Can code be reused or reuse something else?	N	Missing abstraction; Rocket.java tightly coupled.
Minimal use of global variables?	N	Public fields exist (D-017, D-022).
Are related functions grouped properly?	N	Cohesion violated in Rocket.java.
Is the code portable?	Y	Java platform independent.
Are specific types used (int32, unsigned, etc.)?	N	Generic int types used.
Are nested if/else structures limited to 2 deep?	Y	Within acceptable range.
Are nested switch statements avoided?	Y	None found.

Arithmetic Operations:

Description of Item	Pass(Y/N)	Comments
Avoid comparing floating-point numbers for equality?	Y	No FP comparison issues found.
Prevent rounding errors?	N/A	Not applicable.
Avoid additions/subtractions with large magnitude differences?	N/A	Not relevant.
Are divisors tested for zero or noise?	N	Divide by zero error (D-001).

Loops and Branches:

Description of Item	Pass(Y/N)	Comments
Are loops and branches complete and properly nested?	N	Improper termination in Rocket.java loop.
Are common cases tested first in IF chains?	N	No optimization for common paths.
Are all cases covered in IF/CASE blocks?	N	Missing else/default clauses.
Does every case statement have a default?	N	Some switches lack defaults.
Are loop termination conditions achievable?	N	retry-- > -1 runs extra times.
Are indexes properly initialized before loops?	Y	Yes.
Can statements inside loops move outside?	Y	Some can be moved for efficiency.
Does code manipulate index variable after loop?	N	No misuse observed.

Defensive Programming:

Description of Item	Pass(Y/N)	Comments
Are indexes/pointers tested for bounds?	N	Array Index Out of BoundsException (D-003).
Is input validated for validity and completeness?	N	No input validation.
Are all output variables assigned?	Y	Outputs initialized.
Is correct data used in each statement?	N	Type mismatches found.
Is every memory allocation deallocated?	N	Streams not closed (D-015).
Are timeouts/error traps used for device access?	N/A	No devices used.
Are files checked before access?	N	File Input Stream not checked or closed.
Are files/devices left in correct state on termination?	N	Not handled properly.

Maintainability:

Description of Item	Pass(Y/N)	Comments
Does the code make sense?	N	Logic unclear.
Does it comply with coding conventions?	N	Violations throughout.
Does it follow best practices?	N	SOLID principles broken.
Does it follow comment conventions?	N	Inconsistent and missing comments.
Is commenting clear and adequate?	N	Sparse documentation.

Are ideas presented clearly in the code?	N	Poor readability.
Is encapsulation done properly?	N	Public fields.
Is the code overly complex?	Y	Convoluted logic.
Are there unnecessary global variables?	Y	Public static counters.
Is source code readable top-down?	N	Flow confusing.
Are there unused variables or functions?	Y	Unused object and methods.

Requirements and Functionality:

Description of Item	Pass(Y/N)	Comments
Does code match requirements/specifications?	N	Not fully functional; compile errors.
Is the logic proper and functional?	N	Several runtime and logic bugs.

System and Library Calls:

Description of Item	Pass(Y/N)	Comments
Do all system calls have return status checked?	N	Not checked.
Are errors from system/library calls handled?	N	Exceptions ignored.
Are signals caught and handled?	N/A	Not relevant.
Is mutex used on shared variables?	N/A	No multithreading.

Reusability:

Description of Item	Pass(Y/N)	Comments
Are available libraries used effectively?	N	Custom logic replaces standard utilities.
Are utility methods reused?	N	Code not modular.
Is code generalized for reuse?	N	Too specific to implementation.
Is code a candidate for reuse?	N	Needs major refactor.

Robustness:

Description of Item	Pass(Y/N)	Comments
Are all parameters checked?	N	No validation.
Are error conditions caught?	N	Exceptions swallowed.
Default case in all switch statements?	N	Missing in some.
Is there non-reentrant code in unsafe areas?	N/A	Not applicable.
Is macro usage proper?	N/A	None used.
Any unnecessary optimization hindering maintenance?	N	None, but inefficient logic exists.

Security:

Description of Item	Pass(Y/N)	Comments
Does the code pose a security concern?	Y	Hardcoded secret token.
Are service methods annotated with @Authorize?	N/A	Not applicable.
Is inclusion whitelist used for input validation?	N/A	No user input.
Is all user input encoding set by server?	N/A	Not applicable.
Is character encoding set by server?	N/A	Not applicable.
Are cookies with sensitive data secure?	N/A	Not used.
Are input surfaces validated to prevent XSS/SQLi?	N/A	No web module.
Does design address canonicalization issues?	N/A	Not relevant.

Control Structures:

Description of Item	Pass(Y/N)	Comments
Does the app log sensitive data in plain text?	N	No logs observed.
Sensitive data stored in cookies?	N/A	None.
Is sensitive data stored unencrypted?	Y	Auth token hardcoded.
Is encryption used for transmission?	N/A	No network layer.
Is caching disabled for sensitive data?	N/A	Not applicable.

Is email transfer encrypted?	N/A	Not applicable.
Does code use infinite loops?	N	No infinite loops.
Does loop iterate correct number of times?	N	retry loop runs extra iterations.

Resource Leaks:

Description of Item	Pass(Y/N)	Comments
Does code release resources?	N	FileInputStream left open.
Does code release resources twice?	N	No duplicate releases.
Is most efficient class used for resources?	N	Could use try-with-resources.

Error Handling:

Description of Item	Pass(Y/N)	Comments
Does code follow exception handling conventions?	N	Exceptions ignored.
Does code use exception handling properly?	N	Improper catch and print Stack Trace only.
Does code simply catch and log exceptions?	Y	Only stack trace logged.
Does code catch general Exception?	Y	Catches java.lang.Exception.
Are expected values validated?	N	Missing sanity checks.
Are parameters checked for validity?	N	No null checks.

Are errors propagated correctly?	N	Exceptions swallowed.
Are null pointers handled?	N	Null Pointer Exception risk (D-014).
Do switch statements have defaults?	N	Missing.
Are arrays checked for bounds?	N	Fails at D-003.
Is garbage collection done properly?	N	finalize() misused.
Is overflow/underflow checked?	N	Divide by zero bug.
Are errors logged meaningfully?	N	Generic stack traces only.
Would try/catch be useful?	Y	Yes, needed for risky sections.

Timing:

Description of Item	Pass(Y/N)	Comments
Is worst-case timing bounded?	N	retry loop unbounded.
Any race conditions?	N/A	No threads.
Is thread safety ensured?	N/A	Single-threaded.
Any long-running ISRs?	N/A	Not applicable.
Is priority inversion handled?	N/A	No RTOS.
Is watchdog timer used?	N/A	Not applicable.
Has code readability been sacrificed for optimization?	N	Code unoptimized but readable.

Validation & Test:

Description of Item	Pass(Y/N)	Comments
Is code easy to test?	N	Coupled and complex logic.
Do unit tests have full coverage?	N	No test suite.
Is code warning-free on compile?	N	Syntax/type errors.
Are corner cases tested?	N	No handling for invalid inputs.
Can faulty conditions be injected?	N	No test hooks.
Are all interfaces tested?	N	Missing interface validation.
Is worst-case resource use validated?	N	No profiling.
Are assertions used?	N	None present.
Is commented-out test code removed?	Y	No leftover test comments.

Hardware:

Description of Item	Pass(Y/N)	Comments
Do I/O operations set correct hardware state?	N/A	No hardware control.
Are min/max timing requirements met?	N/A	Not applicable.
Multi-byte register consistency ensured?	N/A	Not applicable.
Does software reset to known state?	N/A	Not applicable.
Are brownouts handled?	N/A	Not applicable.
Is system correctly configured for sleep modes?	N/A	Not applicable.

Unused interrupts directed to handler?	N/A	Not relevant.
EEPROM corruption avoided?	N/A	Not applicable.